

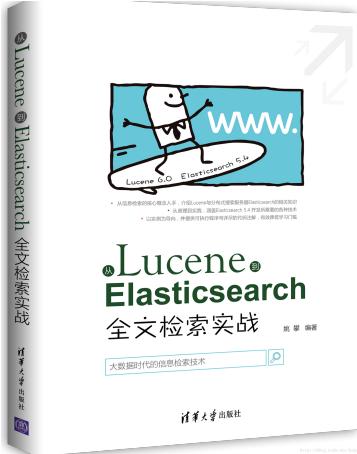
姚攀的博客 1.01^365=31.78

Engineers are versatile minds who create links between science, technology, and society.

[目录视图](#)
[摘要视图](#)
[RSS 订阅](#)

«从Lucene到Elasticsearch：全文检索实战»

希望本书能对您的工作和学习带来帮助，感谢支持！



当当 京东 亚马逊 天猫

Lucene、ES、ELK开发交流群:370734940

[加入QQ群](#)

个人资料



yyyseb

关注

发私信



访问: 844552次

积分: 8642

等级:

排名: 第2540名

图灵赠书——程序员11月书单 【思考】Python这么厉害的原因竟然是！
书和作译者评选启动！ 每周荐书：京东架构、Linux内核、Python全栈
感恩节赠书：《深度学习》等异步社区优秀图书

Spring全家桶(六)必知必会的java注解技术

标签: [java注解](#) [元注解](#) [标准注解](#) [自定义注解](#) [annotation](#)

2017-05-16 11:32

947人阅读

评论(0)

分类:

[spring \(8\)](#)

版权声明: 本文为博主原创文章,未经博主允许禁止转载(<http://blog.csdn.net/napoay>)

[目录\(?\)](#)

[+]

一、Java注解技术的基本概念

Java注解又称Java标注，通俗的说注解就是对某一事物添加注释说明，是Java 5.0版本开始支持加入源代码的特殊语法元数据。Java语言中的类、方法、变量、参数和包都可以被标注，Java标注可以通过反射获取标注内容。在编译器生成类文件时，标注可以嵌入到字节码中。Java虚拟机可以保留标注内容，在运行时可以获取到标注内容。

注解提供了安全的类似注释的机制，用来将任何的信息或元数据与程序元素进行管理。

注解的原理：

注解不会影响程序代码的执行。

二、Java标准注解

@Override

检查该方法是否是重载方法。如果发现其父类，或者是引用的接口中并没有该方法时，会报编译错误。

@Deprecated

标记过时方法。如果使用该方法，会报编译警告。

@SuppressWarnings

指示编译器去忽略注解中声明的警告。

测试，新建一个Person接口，里面定义三个方法，name()和age()用来定义姓名和年龄。

```

1 package com.stuspring.annotation;
2 public interface Person {
3     public String name();
4     public int age();
5     @Deprecated
6     public void sing();
7 }
```

阅读排行

- | | |
|-----------------------------------|---------|
| [搜索]ElasticSearch Java Api(... | (61756) |
| Elasticsearch索引mapping的写... | (36806) |
| Elasticsearch 5.X下JAVA API使... | (31047) |
| 搭建Elasticsearch 5.4分布式集群 | (29631) |
| Elasticsearch 5.1.1 head插件安... | (25822) |
| Elasticsearch java api(五) Bulk... | (21822) |
| ElasticSearch Java Api(四) -删... | (19887) |
| mac命令行启动tomcat | (18615) |
| Elasticsearch 5 Ijk+pinyin分词... | (17900) |
| Elasticsearch shield权限管理详解 | (17369) |

博客专栏



Spring全家桶

文章： 10篇
阅读： 10622



MapReduce编程

文章： 7篇
阅读： 24325



Lucene&Elasticsearch&ELK专栏

文章： 42篇
阅读： 461286



数据库

文章： 9篇
阅读： 18151



J2EE

文章： 24篇
阅读： 92799



ruby on rails

文章： 18篇
阅读： 37381

数据结构与算法

Son类继承Person类：

```

1 package com.stuspring.annotation;
2 public class Son implements Person {
3
4     @Override
5     public String name() {
6         return null;
7     }
8
9     @Override
10    public int age() {
11        return 0;
12    }
13
14    @Override
15    public void sing() {
16    }
17
18 }
```

@Override表明该方法是从父类继承而来的。如果在Person类中删除一个sing()方法，Son类的sing()方法仍然使用@Override来标注编译器就会报错，因为父类中并没有这个方法。

在父类中把sing()方法标注为@Deprecated，表明该方法已经过时，不推荐使用了。调用的时候会给出警告：

The screenshot shows a code editor with Java code. A specific line of code is highlighted with a green background and a yellow border, indicating a warning. The code is as follows:

```

1 package com.stuspring.annotation;
2
3 /**
4  * Created by bee on 17/4/30.
5  */
6 public class Main {
7
8     public static void main(String[] args) {
9         Person p=new Son();
10        p.sing();http://blog.csdn.net/napoay
11    }
12 }
```

如果想忽略警告，可以使用@SuppressWarnings标注：



文章: 13篇
阅读: 17571

文章分类

Elasticsearch (43)
Lucene (15)
hadoop (16)
J2EE (35)
数据结构 (16)
前端 (16)
ruby (18)
机器学习 (4)
python (5)
linux (11)
latex (2)
数据库 (12)
spring (9)
tools (1)
hibernate (1)
mybatis (1)
生活 (1)

文章存档

2017年12月 (9)
2017年11月 (1)
2017年10月 (3)
2017年09月 (2)
2017年08月 (1)

展开

最新评论

MapReduce编程(六) 从HDFS导入数据到Elasticsearch
qq_38094271 :用最新版本的那个成功了。
注意json文件最后一行要换行
《从Lucene到Elasticsearch:全文检索实战...
牙小木 :书已买，正在看
Lucene扩展停用词字典与自定义词库
yyseb :@cuipeng6561:这个文件在ik的目录下。
Lucene扩展停用词字典与自定义词库
cuipeng6561 :IKAnalyzer.cfg.xml 这个文件是要自己写么？

新人千万不要在 Windows 上使用 Ruby on ...
韩大嘴 :对于新手的我来说的，相当中肯！

《从Lucene到Elasticsearch:全文检索实战...
艾鹤 :6666，恭喜发财。

Elasticsearch 5.X下JAVA API使用指南
qq_41296216 :@napoay:maven坐标是 <dependency> ...

Elasticsearch java api(五) Bulk批量索引
yyseb :@wangmaohong0717:可以不指定，会自动生成。

Elasticsearch 5.X下JAVA API使用指南
yyseb :@qq_41296216:贴一下maven坐标吧，或者，博客上有QQ群，加群讨论

```

Main
1 /**
2  * Created by bee on 17/4/30.
3 */
4
5 /**
6  * @SuppressWarnings("deprecation")
7  */
8 public class Main {
9
10    public static void main(String[] args) {
11        Person p=new Son();
12        p.sing();
13    }
14 }

```

三、Java注解分类

按运行机制分，注解分为：

1. 源码注解：注解只在源码中存在，在.class文件中不存在
2. 编译时注解：注解在源码和.class文件中都存在
3. 运行时注解：注解在运行阶段还起作用，甚至会影响程序的运行逻辑。

按照注解来源可以分为：

1. 来自JDK的注解
2. 来自第三方的注解
3. 自定义的注解

*四、Java元注解

元注解是对Java注解进行注解。Java 5.0定义的元注解在

`import java.lang.annotation` 包中，有以下四种类型：

1. `@Target`: 用于描述注解的使用范围，即被描述的注解可以用在什么地方，取值有以下值：
 - CONSTRUCTOR
 - FIELD
 - LOCAL_VARIABLE
 - METHOD
 - PACKAGE
 - PARAMETER
 - TYPE

如下两个注解，它们的作用域分别为方法和域。

```

1 @Target(ElementType.TYPE)
2 public @interface Table{
3     //数据表名称注解， 默认值为类名称
4     public String tableName() default "className";
5 }
6
7

```

Elasticsearch java api(五) Bulk批量索引
hello-friend : 批量索引数据的时候, 必须指定_id吗

```
8 @Target(ElementType.FIELD)
  public @interface NoDBColumn{}
```

1. @Retention

用于描述注解的生命周期。取值有以下几个

SOURCE: 在源文件中有效

CLASS: 在源文件、.class文件中保留

RUNTIME: 在运行时有效

2. @Documented: 标记注解, 标记Documented后javadoc中会生成注解的文档说明

3. @Inherited: 标记注解, 阐述了某个注解的类型是被继承的。使用@Inhe

这个注解将被用于该class的子类。只会集成父类类上的注解, 不会继承子类的注解。

五、Java自定义注解

自定义注解的语法:

```
1 @<注解名>(<成员1>=<成员值1>,<成员1>=<成员值1>)
2
3 @Description(desc="I am eyeColor",author="Mooc boy",age=18)
```

新建一个Description.java, 代码如下:

```
1 package com.stuspring.annotation;
2
3 import java.lang.annotation.*;
4
5 /**
6  * Created by bee on 17/4/30.
7 */
8
9 @Target({ElementType.METHOD})           //作用域 用在方法上
10 @Retention(RetentionPolicy.RUNTIME)    //声明周期
11 @Inherited                           //允许子继承
12 @Documented                          //生成javadoc时会包含注解
13 public @interface Description {
14     String desc(); //成员无参无异常方式声明
15     String author();
16     int age() default 18; // 使用default为成员指定一个默认值
17 }
```

上面代码中定义了一个注解, 注解的名字为Description, 里面有desc、author、age三个成员, 注解作用在方法上, 生命周期为在运行时扔有作用, 允许子继承, 生成javadoc会包含注解。

六、通过反射解析注解

通过反射来获取类、函数或成员上的运行时注解信息, 从而实现动态控制程序运行的逻辑。

首先自定义一个注解,Desc.java

```

1 package com.stuspring.annotation;
2
3 import java.lang.annotation.*;
4
5 /**
6 * Created by bee on 17/5/3.
7 */
8
9 @Target({ElementType.TYPE, ElementType.METHOD})
10 @Retention(RetentionPolicy.RUNTIME)
11 @Inherited
12 @Documented
13 public @interface Desc {
14
15     String value();
16 }
```

在Son.java中使用注解:

```

1 package com.stuspring.annotation;
2
3 /**
4 * Created by bee on 17/4/30.
5 */
6
7 @Desc("I am class annotation")
8 public class Son implements Person {
9
10     @Override
11     @Desc("I am method annotation")
12     public String name() {
13         return null;
14     }
15
16     @Override
17     public int age() {
18         return 0;
19     }
20
21     @Override
22     public void sing() {
23     }
24 }
```

通过反射获取注解的信息, ParseAnn.java

```

1 package com.stuspring.annotation;
2
3 import java.lang.annotation.Annotation;
4 import java.lang.reflect.Method;
```

```

5
6 /**
7 * Created by bee on 17/5/3.
8 */
9 public class ParseAnn {
10     public static void main(String[] args) {
11         //1.使用类加载器加载类
12         try {
13             Class c = Class.forName("com.stuspring.annotation.Son");
14             //2.找到类上面的注解
15             boolean isExist = c.isAnnotationPresent(Desc.class);
16
17             if (isExist) {
18                 //3.拿到注解实例
19                 Desc desc = (Desc) c.getAnnotation(Desc.class);
20                 //4.打印注解的值
21                 System.out.println(desc.value());
22             }
23
24             //5.找到方法上的注解
25             Method[] ms = c.getMethods(); //得到类上的所有方法
26             for (Method m : ms) { //遍历
27                 boolean isExistMethod = m.isAnnotationPresent(Desc.class);
28                 if (isExistMethod) {
29
30                     Desc desc = m.getAnnotation(Desc.class);
31                     System.out.println(desc.value());
32                 }
33             }
34
35             //6.另一种解析方法
36
37             for (Method m:ms){
38                 Annotation[] anns=m.getAnnotations();
39                 for (Annotation a:annts){
40                     if (a instanceof Desc){
41                         Desc desc= (Desc) a;
42                         System.out.println(desc.value());
43                     }
44                 }
45             }
46         } catch (ClassNotFoundException e) {
47             e.printStackTrace();
48         }
49
50     }
51 }

```

运行结果:

```

1 I am class annotation
2 I am method annotation
3 I am method annotation

```

七、注解项目实战

项目说明：

实现一个持久层框架，用来代替Hibernate的解决方案，核心代码是通过注解来实现。

需求：

- 1.有一张用户表，包括用户ID、用户名、昵称、年龄、性别、所在城市、邮箱、手机号
- 2.方便的对每个字段或字段的组合条件进行检索，打印出SQL。
- 3.使用方式要足够简单

Filter类：

```

1 package com.stuspring.annotation.inaction;
2
3 /**
4 * Created by bee on 17/5/3.
5 */
6
7
8 @Table("user")
9 public class Filter {
10
11     @Column("id")
12     private int id;
13
14     @Column("userName")
15     private String userName;
16
17     @Column("nickName")
18     private String nickName;
19
20     @Column("age")
21     private int age;
22
23     @Column("city")
24     private String city;
25
26     @Column("email")
27     private String email;
28
29     @Column("mobile")
30     private String mobile;
31
32
33     public void setId(int id) {
34         this.id = id;
35     }
36
37     public void setUserName(String userName) {
38         this.userName = userName;
39     }
40
41     public void setNickName(String nickName) {

```

```

42         this.nickName = nickName;
43     }
44
45     public void setAge(int age) {
46         this.age = age;
47     }
48
49     public void setCity(String city) {
50         this.city = city;
51     }
52
53     public void setEmail(String email) {
54         this.email = email;
55     }
56
57     public void setMobile(String mobile) {
58         this.mobile = mobile;
59     }
60
61
62     public int getId() {
63         return id;
64     }
65
66     public String getUserName() {
67         return userName;
68     }
69
70     public String getNickName() {
71         return nickName;
72     }
73
74     public int getAge() {
75         return age;
76     }
77
78     public String getCity() {
79         return city;
80     }
81
82     public String getEmail() {
83         return email;
84     }
85
86     public String getMobile() {
87         return mobile;
88     }
89
90
91 }

```

Table注解

```

1 package com.stuspring.annotation.inaction;
2

```

```

3 import java.lang.annotation.ElementType;
4 import java.lang.annotation.Retention;
5 import java.lang.annotation.RetentionPolicy;
6 import java.lang.annotation.Target;
7
8 /**
9 * Created by bee on 17/5/3.
10 */
11
12
13 @Target({ElementType.TYPE})
14 @Retention(RetentionPolicy.RUNTIME)
15 public @interface Table {
16
17     String value();
18 }

```

Column注解:

```

1 package com.stuspring.annotation.inaction;
2
3 import java.lang.annotation.ElementType;
4 import java.lang.annotation.Retention;
5 import java.lang.annotation.RetentionPolicy;
6 import java.lang.annotation.Target;
7
8 /**
9 * Created by bee on 17/5/3.
10 */
11
12 @Target({ElementType.FIELD})
13 @Retention(RetentionPolicy.RUNTIME)
14 public @interface Column {
15
16     String value();
17 }

```

测试:

```

1 package com.stuspring.annotation.inaction;
2
3 import java.lang.reflect.Field;
4 import java.lang.reflect.Method;
5
6 /**
7 * Created by bee on 17/5/3.
8 */
9 public class Test {
10
11     public static void main(String[] args) {
12
13         Filter f1 = new Filter();
14         f1.setId(10);
15

```

```
16 Filter f2 = new Filter();
17 f2.setUserName("lucy");
18
19 Filter f3 = new Filter();
20 f3.setEmail("liu@sina.com,zh@163.com,1235@qq.com");
21
22
23 Filter f4 = new Filter();
24 f4.setEmail("liu@sina.com");
25
26 Filter f5 = new Filter();
27 f5.setUserName("Tom");
28 f5.setAge(20);
29
30     String sq1 = query(f1);
31     String sq2 = query(f2);
32     String sq3 = query(f3);
33     String sq4 = query(f4);
34     String sq5 = query(f5);
35     System.out.println(sq1);
36     System.out.println(sq2);
37     System.out.println(sq3);
38     System.out.println(sq4);
39     System.out.println(sq5);
40
41 }
42
43 public static String query(Filter filter) {
44
45     StringBuilder sb = new StringBuilder();
46     //1.获取到Class
47     Class c = filter.getClass();
48     //获取到Table的名字
49     boolean exist = c.isAnnotationPresent(Table.class);
50     if (!exist) {
51         return null;
52     }
53
54     Table table = (Table) c.getAnnotation(Table.class);
55     String tbName = table.value();
56     sb.append("SELECT * FROM ").append(tbName).append(" WHERE 1=1 ");
57     //遍历所有字段
58     Field[] fArray = c.getDeclaredFields();
59     for (Field field : fArray) {
60         boolean fExists = field.isAnnotationPresent(Column.class);
61         if (!fExists) {
62             continue;
63         }
64
65         Column column = field.getAnnotation(Column.class);
66         String columnName = column.value();
67         //拿到字段的值
68         String fieldName = field.getName();
69         Object fieldValue = null;
70
71         String getMethodName = "get" + fieldName.substring(0, 1).toUpperCase() + fieldName.substring(1);
```

```

72         try {
73             Method getMethod = c.getMethod(getMethodName);
74             fieldValue = getMethod.invoke(filter, null);
75         } catch (Exception e) {
76             e.printStackTrace();
77         }
78
79         //拼装SQL
80
81         if (fieldValue == null || (fieldValue instanceof Integer && (Integer
82             continue;
83         }
84
85         if (fieldValue instanceof String) {
86
87             if (((String) fieldValue).contains(",")) {
88                 String[] values = ((String) fieldValue).split(
89                     sb.append("in(");
90                     for (String v : values) {
91                         sb.append("\'").append(v).append("\'").append(",");
92                     }
93                     sb.deleteCharAt(sb.length() - 1);
94                     sb.append(")");
95             } else {
96                 fieldValue = "\'" + fieldValue + "\'";
97                 sb.append(" and ").append(fieldName).append(" = ").append(
98                     );
99
100            }else{
101                sb.append(" and ").append(fieldName).append(" = ").append(field
102                    );
103
104            }
105            return sb.toString();
106        }
107    }

```

输出:

```

1 SELECT * FROM user WHERE 1=1 and id = 10
2 SELECT * FROM user WHERE 1=1 and userName = 'lucy'
3 SELECT * FROM user WHERE 1=1 in('liu@sina.com','zh@163.com','1235@qq.com')
4 SELECT * FROM user WHERE 1=1 and email = 'liu@sina.com'
5 SELECT * FROM user WHERE 1=1 and userName = 'Tom' and age = 20

```

顶 踩
1 0

- 上一篇 Spring全家桶(五)Bean的多种配置方法
- 下一篇 Spring全家桶(七)通过注解配置Bean

相关文章推荐

- Java必知必会：异常机制详解
- MySQL在微信支付下的高可用运营--莫晓东
- 《Java虚拟机》必知必会——十四个问题总结（内存...）
- 容器技术在58同城的实践--姚远
- 图解 & 深入浅出Java初始化与清理：构造器必知...
- SDCC 2017之容器技术实战线上峰会
- java反射必知必会
- SDCC 2017之数据库技术实战线上峰会
- 《Java虚拟机》必知必会的 14 个问题总结（内存...
- 腾讯云容器服务架构实现介绍--董晓杰
- 《Java虚拟机》必知必会——十四个问题总结（内...
- 微博热点事件背后的数据库运维心得--张冬洪
- 深入理解Spring 之 Spring 进阶开发必知必会 之 Spr...
- Java工程师必知必会的！
- 移动开发必知必会的六大数据统计：
- mysql必知必会（六）

查看评论

暂无评论

您还没有登录,请[\[登录\]](#)或[\[注册\]](#)

* 以上用户言论只代表其个人观点, 不代表CSDN网站的观点或立场

[公司简介](#) | [招贤纳士](#) | [广告服务](#) | [联系方式](#) | [版权声明](#) | [法律顾问](#) | [问题报告](#) | [合作伙伴](#) | [论坛反馈](#)

[网站客服](#) [杂志客服](#) [微博客服](#) webmaster@csdn.net 400-660-0108 | 北京创新乐知信息技术有限公司 版权所有 | 江苏知之为计算机有限公司 |

江苏乐知网络技术有限公司

京 ICP 证 09002463 号 | Copyright © 1999-2017, CSDN.NET, All Rights Reserved 